*electronics*

MDPI

*Article*

# Enhancing Model Agnostic Meta-Learning via Gradient Similarity Loss

**Jae-Ho Tak and Byung-Woo Hong ***

Department of Artificial Intelligence, Chung-Ang University, Seoul 156-756, Republic of Korea; jaeho@image.cau.ac.kr
* Correspondence: hong@cau.ac.kr

**Abstract:** Artificial intelligence (AI) technology has advanced significantly, now capable of performing tasks previously believed to be exclusive to skilled humans. However, AI models, in contrast to humans who can develop skills with relatively less data, often require substantial amounts of data to emulate human cognitive abilities in specific areas. In situations where adequate pre-training data is not available, meta-learning becomes a crucial method for enhancing generalization. The Model Agnostic Meta-Learning (MAML) algorithm, which employs second-order derivative calculations to fine-tune initial parameters for better starting points, plays a pivotal role in this area. However, the computational demand of this method can be challenging for modern models with a large number of parameters. The concept of the Approximate Hessian Effect is introduced in this context, examining the effectiveness of second-order derivatives in identifying initial parameters conducive to high generalization performance. The study suggests the use of cosine similarity and squared error (L2 loss) as a loss function within the Approximate Hessian Effect framework to modify gradient weights, aiming for more generalizable model parameters. Additionally, an algorithm that relies on first-order calculations is presented, designed to achieve performance levels comparable to MAML. This approach was tested and compared with traditional MAML methods using both the MiniImagenet dataset and a modified MNIST dataset. The results were analyzed to evaluate its efficiency. Compared to previous studies that achieved good performance using only the first derivative, this approach is more efficient because it does not require iterative loops to converge on additional loss functions. Additionally, there is potential for further performance enhancement through hyperparameter tuning.

**Keywords:** cosine similarity; deep learning; few-shot image classification; gradient-based meta-learning; model agnostic meta-learning; second-order gradient



**Citation:** Tak, J.-H.; Hong, B.-W. Enhancing Model Agnostic Meta-Learning via Gradient Similarity Loss. *Electronics* **2024**, *13*, 535. https://doi.org/10.3390/electronics13030535

Academic Editor: Ping-Feng Pai

Received: 7 December 2023
Revised: 15 January 2024
Accepted: 26 January 2024
Published: 29 January 2024



## 1. Introduction

Meta-learning, often termed as "learning to learn", is a potent technique that facilitates rapid learning from limited data [1–3]. This approach encompasses learning from a collection of sample tasks, with each task representing a distinct learning problem. The primary objective is to glean knowledge from these tasks, thereby expediting the learning process for new tasks, even with minimal samples. Broadly, meta-learning methodologies can be segmented into metric-based, memory-based, and optimization-based techniques [4–7]. Among these, optimization-based methods are universally applicable and have been efficaciously employed across diverse applications [8]. Parallelly, the paradigm of pre-training followed by fine-tuning has gained traction in the realm of neural networks, predominantly for transfer learning and semi-supervised learning [9,10]. This approach has led to enhanced model generalization and adaptability in various domains, including computer vision, natural language processing [11–13], and the realm of reinforcement learning [14].

Modern deep learning models possess a large number of parameters, which can be advantageous for leveraging vast amounts of training data. However, when the amount of training data is limited, the risk of overfitting increases, particularly when the number

of meta-parameters is relatively high [15]. In such cases, effective meta-learning algorithms that are well-suited to deep learning models become essential for efficiently training high-performance models in scenarios where sufficient pre-training data is not available. MAML is a leading optimization-based approach that aims to estimate optimal parameter initialization for rapid adaptation to new tasks [8]. However, MAML requires the computation of second-order derivatives due to the meta-loss derivative with respect to the initial parameters. This can be prohibitively computationally expensive for large networks. To address this issue, first-order approximations have been developed, such as FOMAML and Reptile, but their convergence and generalization behavior remain underexplored [16,17]. Furthermore, in some applications such as few-shot classification on the Omniglot and tieredImageNet datasets, first-order approximation approaches have been shown to suffer from generalization performance degradation [18].

We consider the role of MAML's second derivatives in enhancing generalization performance and hypothesize that the role of the terms involving second-order derivatives is to determine the weights of the gradient components computed from the query set. Our approach does not exclude second-order derivatives or use alternative methods to improve generalization performance, but rather replaces second-order derivative information through approximation, resulting in performance comparable to that of MAML. To be more specific, we utilized the Approximate Hessian Effect (AHE) to generate the expected effect of second-order derivatives derived from the support set. AHE is weights that approximate the inner loop gradient ratio through calculation of cosine similarity loss and L2 loss. To find the optimal initial parameters, the meta-gradient is multiplied by AHE weights, which can mimic the positive effects brought by second-order derivatives. This allows for the substitution of computationally infeasible operations in models with vast parameters using only first-order derivative information gathered from the inner loop. This enhances the feasibility of applying meta-learning in cutting-edge models, expanding its applicability in the latest models.

## 2. Related Works

In this section, we will discuss Model Agnostic Meta-Learning (MAML), a framework applicable to all deep learning architectures, and focus particularly on Optimization-based First-Order Meta-Learning, which closely aligns with our research. Additionally, we will introduce cosine similarity loss, a core component of our study and a key term in the field of meta-learning.

### *2.1. Model Agnostic Meta-Learning*

In the evolving landscape of machine learning, the concept of meta-learning, or "learning to learn", has emerged as a pivotal paradigm. The primary objective of meta-learning is to train models on a variety of tasks such that they can swiftly adapt to new, previously unseen tasks with minimal data. One of the most prominent and versatile approaches in this domain is the Model Agnostic Meta-Learning (MAML) algorithm [8]. MAML's strength lies in its ability to be applied to any model trained with gradient descent, making it universally adaptable. The core idea is to find a model initialization that is not only suitable for a specific task but can be fine-tuned rapidly for new tasks. These advantages of MAML are particularly utilized in fields such as medical image classification and segmentation, where data is scarce but well-refined initial parameters are crucial [11,19]. Generally, the fast-adapted parameter $\theta$ is computed from randomly sampled tasks $\{\mathcal{T}_i\}_{i=1}^{\mathcal{B}} \sim p(\mathcal{T})$. The notation of the equations is in Table 1. Each parameter is updated by a single support set to find adaptation parameters as follows:

$$\theta_{\mathcal{T}_i} = \theta - \alpha \nabla_\theta \mathcal{L}(\theta;\ x_i, y_i). \tag{1}$$

For the sake of concise notation, we present a single gradient update in the following equation. However, in practice, multiple gradient updates are often required. While it is feasible to induce adapted on a single set with minimal updates, a single update is typically insufficient. And the meta-loss becomes $\sum_{i=1}^{\mathcal{B}} \mathcal{L}(\theta_{\mathcal{T}_i};\ \tilde{x}_i, \tilde{y}_i)$. $\tilde{x}_i, \tilde{y}_i$ is a query set of

task $\mathcal{T}_i$ and they are a disjoint set with support set. Finally the $\theta$ update with meta-loss is represented as

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{i=1}^{\mathcal{B}} \mathcal{L}(\theta_{\mathcal{T}_i};\ \tilde{x}_i, \tilde{y}_i), \tag{2}$$

and this process is shown in Algorithm 1.

**Table 1.** Technical terms and their definitions.

| | |
|---|---|
| $p(\mathcal{T})$ | Task distribution |
| $\alpha, \beta$ | Learning rate of inner, outer loop |
| $\mathcal{K}$ | Inner loop update iteration number |
| $\mathcal{B}$ | Meta-batch size of outer loop for $\theta$ |
| $\mathbf{g}_k$ | The gradient of inner loop loss from $\theta^k$ |
| $\mathcal{W}_{AHE}$ | The variables approximate $\tilde{\mathcal{H}}$ from $\mathbf{g}_k$ and $\mathbf{g}_{k+1}$ |
| $\varepsilon$ | Defined step size of $\mathbf{g}_{k+\varepsilon}$ |
| $\tilde{\varepsilon}$ | The interpolation rate in (0,1] |
| $\eta$ | Learning rate of $\mathcal{W}_{AHE}$ |

**Algorithm 1** Model Agnostic Meta-Learning [8]

**Require:** $p(\mathcal{T}), \alpha, \beta, \mathcal{K}, \mathcal{B}$
Initialize $\theta$ randomly
**while** not done **do**
  Sample $\mathcal{B}$ batch of tasks $\mathcal{T}_i$ from $p(\mathcal{T})$
  **for** all $\mathcal{T}_i$ **do**
    $\theta^0 = \theta$
    **for** $k = 1$ to $\mathcal{K}$ **do**
      $\theta^k_{\mathcal{T}_i} = \theta^{k-1}_{\mathcal{T}_i} - \alpha \nabla_{\theta^{k-1}_{\mathcal{T}_i}} \mathcal{L}(\theta^{k-1}_{\mathcal{T}_i};\ x_i, y_i)$
    **end for**
    **if** First-Order MAML **then**
      $\mathbf{g}^{\mathcal{T}_i} = \nabla_{\theta^{\mathcal{K}}_{\mathcal{T}_i}} \mathcal{L}(\theta^{\mathcal{K}}_{\mathcal{T}_i};\ \tilde{x}_i, \tilde{y}_i)$
    **else**
      $\mathbf{g}^{\mathcal{T}_i} = \nabla_\theta \mathcal{L}(\theta^{\mathcal{K}}_{\mathcal{T}_i};\ \tilde{x}_i, \tilde{y}_i)$
    **end if**
  **end for**
  $\mathbf{g}_{meta} = \sum_{i=1}^{\mathcal{B}} \mathbf{g}^{\mathcal{T}_i}$
  Update $\theta \leftarrow \theta - \beta \cdot \frac{1}{\mathcal{B}} \cdot \mathbf{g}_{meta}$
**end while**

### *2.2. Optimization-Based First-Order Meta-Learning*

In contemporary deep learning research, the parameter count in leading models has surged. As we venture into applying meta-learning for swift adaptation to novel tasks in such scenarios, the significance of computational efficiency becomes paramount. First-order meta-learning algorithms emerge as a beacon of hope, promising computational efficiency and scalability. Noteworthy methods in this domain include First-Order MAML (FOMAML) [8], which sidesteps the second-order derivatives inherent in MAML. Other methods employ proximal regularization for first-order meta-learning strategies [20–22]. For instance, Reptile [17] exclusively utilizes the derivative of proximal regularization as its gradient, orchestrating an interpolation between adaptation and meta-parameters with the following equation.

$$\begin{aligned} \theta_{\mathcal{T}_i} &= \theta - \alpha \nabla_\theta \mathcal{L}(\theta;\ x_i, y_i) \\ \theta &\leftarrow \theta + \beta \sum_{i=1}^{\mathcal{B}} (\theta_{\mathcal{T}_i} - \theta). \end{aligned} \tag{3}$$

This equation presents a single gradient update in the inner loop for simplicity of notation, similar to (1). As evident from the equation, in contrast to the MAML method, Reptile does not necessitate the division of training data into support and query sets. This confers a notable advantage in terms of data efficiency [23]. Building upon Reptile, Meta-MinibatchProx [21] applies proximal regularization to the updates of adaptation parameters. Upon completing the inner loop using the $\epsilon_s$-approximation, the meta-parameters are updated with the adaptation parameters in a manner consistent with Reptile. Recent research, such as Sign-MAML [24], tackle meta-learning challenges using first-order techniques, leveraging sign-based optimization strategies. Among the studies introduced earlier, those that achieve or surpass the performance of MAML, which include the computation of second-order derivatives, require the loss function to converge to a small pre-defined value at each update [20,21]. This approach could significantly amplify training times and, in some cases, may not be Model Agnostic. In contrast, our study adopts a method of tuning the final meta-gradient based on gradients during the adaptation process, thereby eliminating the need to designate a small value and preventing an increase in training time. The training duration can be compared with that of MAML in Section 5.

### *2.3. Cosine Similarity Loss*

In the domain of deep learning, the adoption of cosine similarity as a loss function has been increasingly recognized, particularly for tasks necessitating the quantification of similarity or distance between feature representations. The loss function is defined as the negative cosine similarity, a formulation chosen because the cosine achieves its maximum value when the similarity is high. The function is mathematically formulated as

$$\mathcal{L}_{\cos}(a, b) = \frac{-a \cdot b}{\|a\|_2 \times \|b\|_2}. \tag{4}$$

The power of cosine similarity lies in its ability to capture the angular distance between vectors, making it particularly suited for normalized feature vectors. One notable application is in the classification of plasma images from the Caltech Spheromak Experiment. In this context, cosine similarity was employed not only for feature selection but also as a loss function for training an AlexNet-based model for plasma image classification [25]. The results demonstrated the efficacy of cosine similarity in distinguishing between unstable and stable columns with high accuracy. Further, the challenges of incremental learning, particularly the catastrophic forgetting problem, have been addressed using a prototype sampling mechanism based on *K*-means clustering. In this approach, cosine similarity plays a pivotal role in enhancing the discrimination between old and new classes. Specifically, a mask attached to the loss function, based on the cosine similarity between prototypes and current data, has been proposed to further refine the distinction compared to traditional knowledge distillation schemes [26]. Such innovative applications underscore the versatility and potential of cosine similarity as a loss function in deep learning paradigms.

## 3. Preliminaries

In this section, we expand the formula of the second-order computation in MAML and explore its significance in identifying fine-tuned initial parameters. Additionally, we will describe how a collection of first-order gradient computations can effectively approximate the form of second-order derivatives.

### *3.1. Second-Order Computation from MAML*

The fast-adapted parameters are expected to be close to the optimal parameters. As a result, the length of the inner loop, denoted as $\mathcal{K}$, is typically set to a value greater than 1,

ensuring that $\theta$ aligns with $\theta^0_{\mathcal{T}_i}$. Utilizing the chain rule, the gradient of the meta-parameters, represented as $\nabla_\theta \sum_{i=1}^{\mathcal{B}} \mathcal{L}(\theta_{\mathcal{T}_i};\ \tilde{x}_i, \tilde{y}_i)$ in (2), can be articulated as

$$\sum_{i=1}^{\mathcal{B}} \frac{\partial \mathcal{L}(\theta^{\mathcal{K}}_{\mathcal{T}_i};\ \tilde{x}_i, \tilde{y}_i)}{\partial \theta^1_{\mathcal{T}_i}} \cdot \frac{\partial \theta^1_{\mathcal{T}_i}}{\partial \theta}. \tag{5}$$

Continuing with the application of the chain rule, the inner loop computation in the equation can be further expanded to

$$\sum_{i=1}^{\mathcal{B}} \frac{\partial \mathcal{L}(\theta^{\mathcal{K}}_{\mathcal{T}_i};\ \tilde{x}_i, \tilde{y}_i)}{\partial \theta^{\mathcal{K}}_{\mathcal{T}_i}} \cdot \Pi_{k=1}^{\mathcal{K}} \frac{\partial \theta^k_{\mathcal{T}_i}}{\partial \theta^{k-1}_{\mathcal{T}_i}}. \tag{6}$$

Referring to (1), the relation between $\theta^{k-1}_{\mathcal{T}_i}$ and $\theta^k_{\mathcal{T}_i}$ is established, leading to the final expression of $\mathbf{g}_{meta}$ as

$$\mathbf{g}_{meta} = \sum_{i=1}^{\mathcal{B}} [\nabla_{\theta^{\mathcal{K}}_{\mathcal{T}_i}} \mathcal{L}(\theta^{\mathcal{K}}_{\mathcal{T}_i};\ \tilde{x}_i, \tilde{y}_i) \cdot \Pi_{k=1}^{\mathcal{K}} (\boldsymbol{I} - \alpha \nabla^2_{\theta^{k-1}_{\mathcal{T}_i}} \mathcal{L}(\theta^{k-1}_{\mathcal{T}_i};\ x_i, y_i))]. \tag{7}$$

The results elucidate that the second-order derivative must be computed at every step within the inner loop. This presence of the second-order derivative enables MAML's training to reduce the loss, which assesses generalization post-adaptation phase, from the parameters set prior to the adaptation phase. However, this characteristic poses a computational challenge for MAML, especially as the length of the inner loop increases and the number of model parameters grows. Notably, the computational complexity for the second-order derivative is denoted as $\mathcal{O}(n^2)$, which is considerably higher than the $\mathcal{O}(n)$ complexity associated with the first-order derivative. This stark difference in complexity underscores the potential inefficiencies in MAML when dealing with large-scale models or extended inner loops. The FOMAML approach circumvents the computation of the second-order derivative, opting instead to solely utilize the gradient, denoted as $\nabla_{\theta^{\mathcal{K}}_{\mathcal{T}_i}} \mathcal{L}(\theta^{\mathcal{K}}_{\mathcal{T}_i};\ \tilde{x}_i, \tilde{y}_i)$.

### *3.2. Hessian-Vector Product*

In practical GPU implementations, the computation of the Hessian's $\mathcal{O}(n^2)$ is avoided by employing the Hessian-vector product for MAML operations, effectively circumventing the problems associated with $\mathcal{O}(n^2)$ complexity. The expansion and result of the Hessian-vector product $\mathbf{Hv}$ are given by

$$\begin{aligned} \mathbf{Hv} &= \lim_{r \to 0} \frac{\nabla \mathcal{L}(\theta + rv) - \nabla \mathcal{L}(\theta)}{r} = \frac{\partial}{\partial r} \nabla \mathcal{L}(\theta + rv) \Big|_{r=0} \\ &= \nabla (\nabla \mathcal{L}(\theta)^T v). \end{aligned} \tag{8}$$

Since we are computing the gradient with respect to a scalar value $\nabla \mathcal{L}(\theta)^T v$, we proceed with the operation of the second derivative in the vector as shown in (7). Consequently, the computational complexity only increases by a constant factor, roughly five times that of first-order derivatives, and the memory usage increases by no more than twice per inner loop [27]. While the impact may vary depending on the model or data structure, our experiments indicate that, compared to FOMAML, the memory usage for MAML increases by an additional 50% for each inner loop iteration. This implies that, if MAML includes two inner loop iterations, it would use approximately twice the memory of FOMAML. It is important to note that FOMAML does not require storing graphs for each inner loop iteration, thus maintaining a constant memory usage regardless of the number of iterations.

### *3.3. Setting the Second-Order Term to Zero is Effective*

At its core, this method calculates the gradient concerning the query set using the parameters derived at the end of the inner loop, which then informs the update of the meta-parameters. As can be observed from (7), this treats the second-order term as zero,

$$\mathbf{g}_{meta} = \sum_{i=1}^{\mathcal{B}} \nabla_{\theta_{\mathcal{T}_i}^{\mathcal{K}}} \mathcal{L}(\theta_{\mathcal{T}_i}^{\mathcal{K}};\ \tilde{x}_i, \tilde{y}_i) \cdot \Pi_{k=1}^{\mathcal{K}} \boldsymbol{I}, \tag{9}$$

allowing the result of the $\Pi_{k=1}^{\mathcal{K}} \boldsymbol{I}$ to be simply reduced to one. This approach is justified by the equation below, where the symbol $\coloneqq$ denotes the definition of a term.

$$\begin{aligned} \mathbf{g}_k &\coloneqq \frac{\partial \mathcal{L}(\theta_k)}{\partial \theta_k} \\ \frac{\partial^2 \mathcal{L}(\theta_k)}{\partial \theta_k^2} &\approx \lim_{\varepsilon \to 0} \frac{\mathbf{g}_{k+\varepsilon} - \mathbf{g}_k}{-\varepsilon \cdot \mathbf{g}_k} \\ &= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \cdot (1 - \frac{\mathbf{g}_{k+\varepsilon}}{\mathbf{g}_k}). \end{aligned} \tag{10}$$

The negative sign in the denominator of the middle formula among the three is due to the update in gradient descent proceeding in the negative direction of the gradient. From the given equation, it is clear that FOMAML approximates the ratio between $\mathbf{g}_{k+\varepsilon}$ and $\mathbf{g}_k$ to 1, while disregarding the fact that inverse of $\varepsilon$ is infinite. Although this methodology might exhibit commendable performance on simpler data structures with an appropriate length of inner loops, it encounters potential pitfalls as the length of these loops increases. The connection between the model parameters, determined at the conclusion of the inner loops, and the meta-parameters becomes tenuous. This can lead to a performance that is markedly subpar compared to MAML. Furthermore, even with a shorter length of inner loops, the meta-gradient may lack the requisite information to effectively mitigate overfitting during the test phase [20]. It is essential to consider these nuances when adopting such an approach, especially in complex scenarios where the integrity of the model's adaptability is paramount. In prior research, the efficacy of FOMAML and Reptile as algorithms for optimization-based meta-learning was analyzed mathematically in comparison with MAML [17]. Consequently, various methods have been explored to effectively substitute the second-order derivative computation [21,28].

## 4. Method

In this section, we elucidate the process of determining the value of the Approximate Hessian Effect (AHE) using the cosine similarity loss. We further illustrate how to transform the gradient for enhanced effectiveness.

### *4.1. Approximate Hessian Effect*

We hypothesize that the role of the terms involving second-order derivatives is to determine the weights of the gradient components computed from the query set. Observing the term $(\boldsymbol{I} - \alpha \nabla_\theta^2 \mathcal{L}(\theta))$, when the second-order derivative is negative, the weight of the corresponding gradient component increases and, when the second-order derivative is positive, the weight of the corresponding gradient component decreases. In deep learning, it is generally assumed that the loss landscape exhibits both nonconvex and convex characteristics. This can be succinctly visualized in Figure 1. Consequently, when the second-order derivative possesses a large positive value, it can be inferred that the corresponding gradient component is already in close proximity to the optimal point. On the other hand, a negative second-order derivative could serve as evidence that the component points are still at a considerable distance from the optimal point. Meta-learning aims to rapidly adapt to a given task while simultaneously avoiding overfitting to the optimal parameters. Therefore, we infer that the term $(\boldsymbol{I} - \alpha \nabla_\theta^2 \mathcal{L}(\theta))$ fine-tunes the gradient de-

rived from the query set to be more effective for generalization. Based on this analysis, we propose a heuristic function to approximate second-order derivatives accordingly. In addition to FOMAML and Reptile, other papers have also explored optimization-based meta-learning using only first-order gradient information [29]. Our approach does not exclude second-order derivatives or use alternative methods to improve generalization performance, but rather replaces second-order derivative information through approximation, resulting in performance comparable to that of MAML. The following equation illustrates the computation of the meta-gradient approximation, denoted as $\tilde{\mathbf{g}}_{\mathbf{meta}}$ using AHE. Our method creates a weight map for fine-tuning the final gradient, enabling more effective updates. To be more specific, we utilized the AHE to generate the expected effect of second-order derivatives derived from the support set. Based on (10), estimate AHE to

$$\begin{aligned} \mathcal{W}_{AHE} &\approx \frac{\mathbf{g}_{k+\varepsilon}}{\mathbf{g}_k} \\ \tilde{\mathcal{H}} &\coloneqq \frac{1}{\varepsilon} \cdot (1 - \mathcal{W}_{AHE}) \\ \tilde{\mathbf{g}}_{\mathbf{meta}} &\coloneqq \sum_{i=1}^{\mathcal{B}} \nabla_{\theta_{T_i}^{\mathcal{K}}} \mathcal{L}(\theta_{T_i}^{\mathcal{K}};\, \tilde{x}_i, \tilde{y}_i) \cdot \Pi_{k=1}^{\mathcal{K}} (\boldsymbol{I} - \alpha \cdot \tilde{\mathcal{H}}^k), \end{aligned} \tag{11}$$

and it differs from FOMAML fixing Hessian to zero; $\mathcal{W}_{AHE}$ find an appropriate value of Hessian. AHE is a variable that takes the input set of $\nabla_\theta \mathcal{L}(\theta;\, x_i, y_i)$ and outputs $\tilde{\mathcal{H}}$. The method for updating and determining the appropriate $\mathcal{W}_{AHE}$ will be elaborated upon in the next section. Additionally, this scenario posits that a function $f : \mathbb{R}^n \to \mathbb{R}$ exhibiting $\mu$-strong convexity satisfies the Hessian condition $\mu \leq \nabla^2 f(x) \leq LI$ for all $x \in \mathbb{R}^n$, where $L$ represents the Lipschitz constant and $I$ denotes the identity matrix. In the context of meta-learning, where tasks with limited data undergo inner loop training, there is an elevated risk of overfitting. Consequently, it is typical for both $\mu$ and $L$ to assume larger values. This implies that all eigenvalues of the Hessian $\nabla^2 f(x)$ are bounded below by $\mu$ and above by $L$. Such characteristics are anticipated to be advantageous in meta-learning settings, as they provide a balanced approach to addressing overfitting while facilitating efficient inner loop learning.

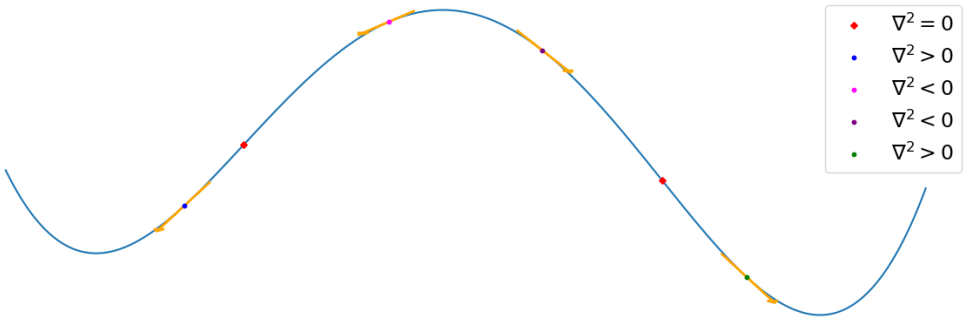



**Figure 1.** In the optimization process, the likelihood of the second derivative taking a positive value increases as it approaches the overfitting-prone optimal point. Viewed from another perspective, if the second derivative is negative, one can anticipate that there is a relative distance to the optimal point, with a possible inflection in between. Given that the support set used in meta-learning is susceptible to overfitting, the term $(I - \alpha \nabla_\theta^2 \mathcal{L}(\theta))$ in MAML is particularly significant. When this term multiplies the gradient obtained from the query set, it can guide the learning in a direction opposite to that which is vulnerable to overfitting during support set training. This suggests that, while the second derivative in MAML gets closer for individual tasks, it does not overfit. This characteristic potentially allows the algorithm to find parameters that can quickly adapt to new tasks without overfitting.

### *4.2. How to Update Variables via Gradient Similarity Loss*

In this study, we incorporated AHE by updating based on the set of inner loop gradients and their associated gradient similarity loss. The fundamental concept revolves around updating $\mathcal{W}_{AHE}$ to align with the ratio of sequential gradients, leveraging both cosine similarity and L2 loss.

#### 4.2.1. Gradient Similarity Loss

For the gradient similarity loss, we employ two loss functions to ensure that the product $\mathbf{g}_k \cdot \mathcal{W}_{AHE}$ closely approximates $\mathbf{g}_{k+\varepsilon}$. This is crucial because, in most meta-learning inner loops, there is a high likelihood that the parameters become overfitted to the support set. Consequently, the gradient may assume exceedingly small values, leading to potential floating-point errors when the gradient is in the denominator. This scenario hinders the accurate approximation of $\mathcal{W}_{AHE}$ to the ratio $\mathbf{g}_{k+\varepsilon}/\mathbf{g}_k$. By interpolation, $\mathbf{g}_{k+\varepsilon}$ is defined as follows

$$\mathbf{g}_{k+\varepsilon} \coloneqq \mathbf{g}_k \cdot (1-\tilde{\varepsilon}) + \mathbf{g}_{k+1} \cdot \tilde{\varepsilon}, \tag{12}$$

and

$$\varepsilon \coloneqq (1-\tilde{\varepsilon}) \cdot \alpha, \tag{13}$$

where $\tilde{\varepsilon}$ denotes the interpolation ratio, progressively decreasing from one to zero during the meta-training phase. For the cosine similarity loss as presented in (4), we normalize the vector to its unit vector because the magnitude of the loss is independent of the vector's scale, yet the gradient remains dependent. Consequently, the cosine similarity loss is formulated as

$$\hat{\mathcal{L}}_{\cos}(a,b) = \frac{-\hat{a} \cdot \hat{b}}{\|\hat{a}\|_2 \times \|\hat{b}\|_2}. \tag{14}$$

The gradient similarity loss, denoted as $\mathcal{L}_{GS}$, is computed across the inner loop as

$$\begin{aligned} \mathcal{L}_{GS}^{k} \coloneqq \sum_{l=1}^{L} [&\hat{\mathcal{L}}_{\cos}(\mathcal{W}_{AHE}^{k,\,l} \cdot \mathbf{g}_k^l, \mathbf{g}_{k+\varepsilon}^l) \\ &+ \lambda \cdot \|\mathcal{W}_{AHE}^{k,\,l} \cdot \mathbf{g}_k^l - \mathbf{g}_{k+\varepsilon}^l\|_2^2], \end{aligned} \tag{15}$$

based on the aforementioned definitions. $\lambda$ represents the coefficient of the L2-norm, while $L$ denotes the number of layers associated with the gradients and parameters. Furthermore, all inputs to the loss functions are vectorized.

#### 4.2.2. MAML via AHE Update

To employ this method within MAML, one can refer to Algorithm 2 where (11) through (15) are comprehensively applied. For the training of $\mathcal{W}_{AHE}$, the gradients resulting from $\mathcal{L}_{GS}$ over the $k$ iterations on the support set are averaged and utilized for training. This mirrors the update seen in the Reptile Algorithm, as depicted in (3). The parameter $\hat{\varepsilon}$ is utilized to adjust the difference between gradients through interpolation, effectively modulating the magnitude of the gradient for AHE, facilitating a more stable approximation. The scheduling of $\hat{\varepsilon}$ may vary depending on the characteristics of the meta-learning problem at hand. A recommended approach is to decrease the value of $\hat{\varepsilon}$ as the potential for overfitting in the inner loop increases, allowing for a larger $\varepsilon$ value as seen in (13). Various scheduling methods, including cosine scheduling that converges to zero before the final outer loop concludes, can be employed. Alternatively, one can directly fix the magnitude of the gradient for AHE based on $\varepsilon$. This approach will be referred to in Section 5.3.

**Algorithm 2** MAML via AHE update

**Require:** $p(\mathcal{T}), \alpha, \beta, \mathcal{K}, \mathcal{B}, \equiv, \hat{\varepsilon}$
Initialize $\theta$ randomly
Initialize $\mathcal{W}_{AHE}$ to 1
**while** not done **do**
  Sample $\mathcal{B}$ batch of tasks $\mathcal{T}_i$ from $p(\mathcal{T})$
  **for** all $\mathcal{T}_i$ **do**
    $\theta^0 = \theta$
    $\mathcal{W}^0_{AHE} = \mathcal{W}_{AHE}$
    **for** $k = 1$ to $\mathcal{K} + 1$ **do**
      $\mathbf{g}_k = \nabla_{\theta^{k-1}_{\mathcal{T}_i}} \mathcal{L}(\theta^{k-1}_{\mathcal{T}_i};\ x_i, y_i)$
      **if** not $k = \mathcal{K} + 1$ **then**
        $\theta^k_{\mathcal{T}_i} = \theta^{k-1}_{\mathcal{T}_i} - \alpha \cdot \mathbf{g}_k$
      **end if**
    **end for**
    **for** $k = 1$ to $\mathcal{K}$ **do**
      $\mathcal{W}^k_{AHE} = \mathcal{W}^{k-1}_{AHE} - \eta \nabla_{\mathcal{W}_{AHE}} \mathcal{L}^k_{GS}$
      $\tilde{\mathcal{H}}^k = \frac{1}{\varepsilon} \cdot (1 - \mathcal{W}^k_{AHE})$
    **end for**
    $\mathbf{g}_{AHE} = \frac{1}{\mathcal{K}} \sum_{k=1}^{\mathcal{K}} \nabla_{\mathcal{W}_{AHE}} \mathcal{L}^k_{GS}$
  **end for**
  $\tilde{\mathbf{g}}_{meta} = \sum_{i=1}^{\mathcal{B}} \nabla_{\theta^{\mathcal{K}}_{\mathcal{T}_i}} \mathcal{L}(\theta^{\mathcal{K}}_{\mathcal{T}_i};\ \tilde{x}_i, \tilde{y}_i) \cdot \Pi_{k=1}^{\mathcal{K}} (I - \alpha \cdot \tilde{\mathcal{H}}^k)$
  Update $\theta \leftarrow \theta - \beta \cdot \frac{1}{\mathcal{B}} \cdot \tilde{\mathbf{g}}_{meta}$
  Update $\mathcal{W}_{AHE} \leftarrow \mathcal{W}_{AHE} - \beta \cdot \frac{\eta}{\mathcal{B}} \cdot \sum_{i=1}^{\mathcal{B}} \mathbf{g}_{AHE}$
**end while**

### *4.3. Comparative Analysis of Proposed Algorithm*

In the realm of meta-learning algorithms, a critical distinction exists between MAML and FOMAML. As discussed in Section 3.2, MAML necessitates second-order operations, specifically the computation of Hessian-vector products. This requirement inherently limits its computational efficiency, especially as the number of inner loop iterations increases, encountering computational constraints. Conversely, while fewer inner loops in MAML may hinder the discovery of robust initial parameters, potentially rendering the process ineffective or impracticable. FOMAML, on the other hand, overlooks the convexity of the loss field, presenting unique challenges. In nonlinear domains, the lack of understanding about the connection between the start and end of an inner loop becomes evident. The reliance solely on the final gradient for updates, as FOMAML employs, leads to inaccuracies. These inaccuracies are exacerbated as the inner loop lengthens. Conversely, shorter inner loops can result in slow or even nonexistent learning, as evidenced in [17]. In contrast, our AHE differs as it does not require the calculation of the Hessian-vector product. The additional complexity introduced in AHE is merely due to the inclusion of an extra first-order derivative calculation, resulting in a computational complexity that is twice as much. However, importantly, the memory usage is not influenced by the number of inner loops. Therefore, AHE can be employed in larger models where the use of MAML is not feasible. Consequently, this approach, even with extended inner loops, continuously integrates the correlation between gradients, thereby adjusting the gradient in the outer loop. This method enables the utilization of FOMAML in a manner similar to MAML. A comparison of computational complexity and memory usage can be referenced in Table 2.

**Table 2.** Comparative analysis of computational complexity among MAML, FOMAML, and AHE, with respect to the inner loop size, denoted as $\mathcal{K}$.

| | **Computational Complexity** | **Space Complexity** |
|---|---|---|
| FOMAML | $\mathcal{C}$ | $\mathcal{S}$ |
| MAML | $\approx 5 \times \mathcal{C}$ | $\approx (1 + 0.5 \times \mathcal{K}) \times \mathcal{S}$ |
| AHE (Ours) | $\approx 2 \times \mathcal{C}$ | $\approx 1.03 \times \mathcal{S}$ |

## 5. Experiments

In this section, we aim to validate the efficacy of our proposed approach by contrasting it with other optimization-based meta-learning techniques in few-shot image classification. We adhere to the experimental protocol established in prior meta-learning research [1].

### *5.1. Implementation Detail*

#### 5.1.1. Dataset

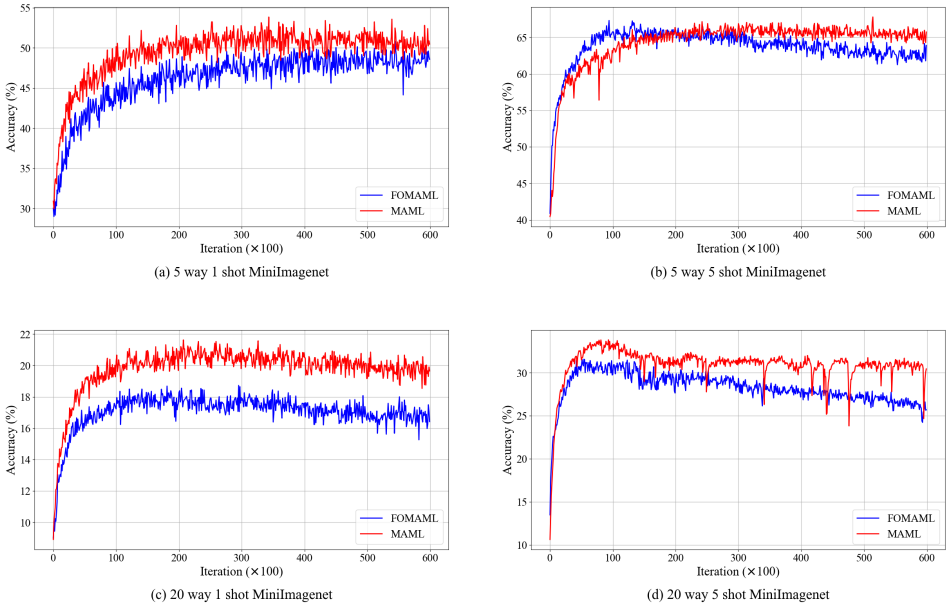
In our experiments, we employ the MiniImagenet dataset, introduced for few-shot image classification [9]. This dataset comprises 64 training classes, 12 validation classes, and 24 test classes. It is frequently used as a benchmark for few-shot learning due to its size and complexity, which are well-suited for both training and testing. Additionally, we conducted experiments on the MNIST dataset with applied modifications. During the training phase, we used the clean MNIST dataset; while in the testing phase, we employed images that contained random levels of noise and were randomly rotated within 90 degrees. The results of these experiments and examples of the test images can be found in Figure 2.


**Figure 2.** (**Left**) MNIST test dataset modified with random noise and rotation. (**Right**) 5-way 1-shot classification test accuracy comparison for MAML and AHE. MAML achieved a top accuracy of 81%, while AHE reached 82%.

#### 5.1.2. Experiment Setting

In the *N*-way *K*-shots classification problem, *N* unseen classes are provided with *K* distinct instances from each of these *N* classes, and the model's proficiency is then tested based on its ability to classify unseen novel instances within these *N* classes. During the training phase, meta-gradients are computed using both the support and query sets, and are subsequently utilized for updates. In the testing phase, only the inner loop update is performed using an unseen test support set. Following this, the model's fast adaptation performance is assessed by measuring accuracy on an unseen test query set. We conducted experiments under four distinct scenarios for 60,000 iterations. For the number of classes in a task, denoted as *N*, we considered both 5 and 20. For the number of data points in each of the support and query sets within a task, denoted as *K*, we used values of 1 and 5. To compare with our method, we conducted experiments with MAML and FOMAML

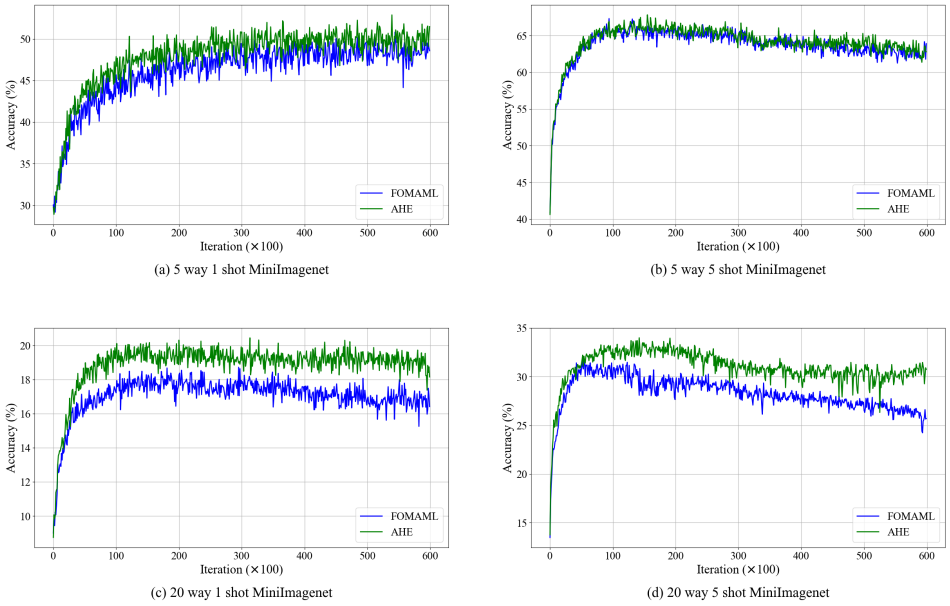
under identical conditions. As can be observed in Figure 3, the peak test accuracies during iterations surpass those previously reported in the research [21]. Consequently, we based our algorithmic comparisons on the accuracies of MAML and FOMAML as tested in our experiments, which can be viewed in Table 3 and Figure 4. Universally, we employed $\alpha = 0.05$, $\beta = 0.001$, and $\mathcal{K} = 10$. For the 5-way 1-shot scenario, we used $\mathcal{B} = 16$, for the 5-way 5-shot scenario $\mathcal{B} = 8$, for the 20-way 1-shot scenario $\mathcal{B} = 8$, and for the 20-way 5-shot scenario $\mathcal{B} = 4$. In all approaches, the Adam optimizer [30] was employed to update the meta-gradient after the outer loop. For the learning of AHE, parameter $\eta$ was set to the reciprocal of $\mathcal{K}$, which is 0.1, and $\lambda$ is 0.1.




(a) 5 way 1 shot MiniImagenet (b) 5 way 5 shot MiniImagenet

(c) 20 way 1 shot MiniImagenet (d) 20 way 5 shot MiniImagenet

**Figure 3.** Experiments on the MiniImagenet dataset using MAML and FOMAML in terms of test accuracy. While (**a**,**c**) represent 1-shot learning, showing MAML significantly outperforming FOMAML from the early stages of training, scenarios like (**b**,**d**) for 5-shot learning depict FOMAML either matching or slightly surpassing MAML in the initial phases. Ultimately, MAML consistently demonstrates superior performance over FOMAML across all cases.




(a) 5 way 1 shot MiniImagenet (b) 5 way 5 shot MiniImagenet

(c) 20 way 1 shot MiniImagenet (d) 20 way 5 shot MiniImagenet

**Figure 4.** Experiments on the MiniImagenet dataset using AHE and FOMAML in terms of test accuracy. In all tests, we observed a slight improvement in the results using AHE.

**Table 3.** Test accuracy (in percent) for few-shot image classification on the MiniImagenet test set. Each value represents the mean across 600 tasks, ± shows 95% confidence intervals over tasks.

| | **5-Way 1-Shot** | **5-Way 5-Shot** | **20-Way 1-Shot** | **20-Way 5-Shot** |
|---|---|---|---|---|
| MAML | $53.83 \pm 1.90$ | $67.79 \pm 0.86$ | $21.62 \pm 0.73$ | $33.82 \pm 0.38$ |
| FOMAML | $50.80 \pm 1.78$ | $67.31 \pm 0.92$ | $18.72 \pm 0.67$ | $31.57 \pm 0.38$ |
| AHE (Ours) | $52.87 \pm 1.90$ | $67.82 \pm 0.87$ | $20.45 \pm 0.65$ | $33.983 \pm 0.37$ |

#### 5.1.3. Evaluation Setup

For the evaluation, we utilized the test classes, conducting tests every 100 training iterations. Throughout all experiments, both the average accuracy and the 95% confidence intervals were derived from the results of 600 tasks. In this stage, the performance of fast adaptation is evaluated based on the accuracy of the test query set, following the parameters which complete the training with the test support set in the inner loop. The test support and query sets maintain the same way and shot configurations as the train support and query sets. In some previous studies, the shot size of the test query set can differ from that of the support set, typically being larger. Consequently, we utilized only $\alpha = 0.05$ and $\mathcal{K} = 10$, consistent with the values applied during the training phase.

#### 5.1.4. Comparison Methods

To ensure a fair comparison of performance, we did not seek the optimal hyperparameters for each problem but instead used a reasonable common value. We used the learn2learn meta-learning library [31]. In Figure 4, we can observe the trend of test accuracy for our method compared to FOMAML. Additionally, the best performances can be compared in Table 3.

### *5.2. Experimental Results*

#### 5.2.1. 5-Way 1-Shot Classification

The 5-way 1-shot scenario is the most straightforward problem, where training is conducted using only one data point per class. This method inherently has a high propensity for overfitting from the early stages of training. Moreover, with a limited set of just five classes, the challenge of elevating fast adaptation performance from the outset is substantial. In the initial phases of training MiniImagenet, MAML displays the most consistent upward trajectory. On the other hand, while AHE exhibits some fluctuations, it generally outperforms FOMAML in terms of accuracy. In the test phase of the experiments on the MNIST dataset with randomly applied noise and rotation, we observed that performance reached over 80% due to training on 10 classes with randomly applied labels during the training phase, which is higher compared to MiniImagenet. However, the modified data in the test phase led to increased vulnerability to overfitting, resulting in a continuous decline in performance even after reaching peak effectiveness. Nonetheless, as Figure 2 illustrates, AHE exhibited a less pronounced performance decline compared to MAML.

#### 5.2.2. 5-Way 5-Shot Classification

In the 5-way 5-shot, which presents the lowest likelihood of overfitting, each class is trained using five data points. Compared to other scenarios, FOMAML significantly outperforms MAML in the early stages of training. However, while MAML achieves a slightly higher peak performance, it notably does not exhibit a performance decline in the later stages, unlike FOMAML. This observation suggests that, in the 5-way 5-shot setting, the parameters are updated to a more generalized state in the later stages of training compared to other scenarios. At this point, the use of second-order derivatives becomes critically important, as hypothesized. AHE, on the other hand, surpasses MAML in terms of peak performance but its decline in the later stages mirrors that of FOMAML.

#### 5.2.3. 20-Way 1-Shot Classification

The 20-way 1-shot scenario is the most susceptible to overfitting. Given that only one data point is provided for each of the 20 classes, overfitting is likely to occur from the outset of the inner loop training. This phenomenon is a primary reason why MAML, which uses second-order derivatives, and AHE, which employs an approximation, exhibit a significant performance gap compared to FOMAML, which relies solely on first-order derivatives. This disparity is evident in Figures 2 and 3. Among all problems, the 20-way 1-shot setting most distinctly highlights the pronounced performance degradation of FOMAML in situations where avoiding overfitting is challenging.

#### 5.2.4. 20-Way 5-Shot Classification

In the 20-way 5-shot scenario, significant fluctuations were observed in both MAML and AHE. While the peak performance for both methods was similar and surpassed that of FOMAML, the pronounced fluctuations that occurred post-peak in the learning curve warrant further investigation. We hypothesize that the presence of numerous classes and shots could lead to a highly intricate nonconvex structure in the convergence region. Consequently, during the learning process, the model might traverse regions that amplify the meta-gradient, leading to a temporary decline in performance. Additionally, we conducted a comparison of the elapsed time between MAML and AHE in the most extensive training model and time-consuming scenario, the 20-way 5-shot. Our findings, as illustrated in Figure 5, show that AHE requires approximately half the time of MAML over 20,000 training loops and 200 testing loops. During these experiments, we utilized a GeForce RTX 3090 with 24 GB of GPU memory.

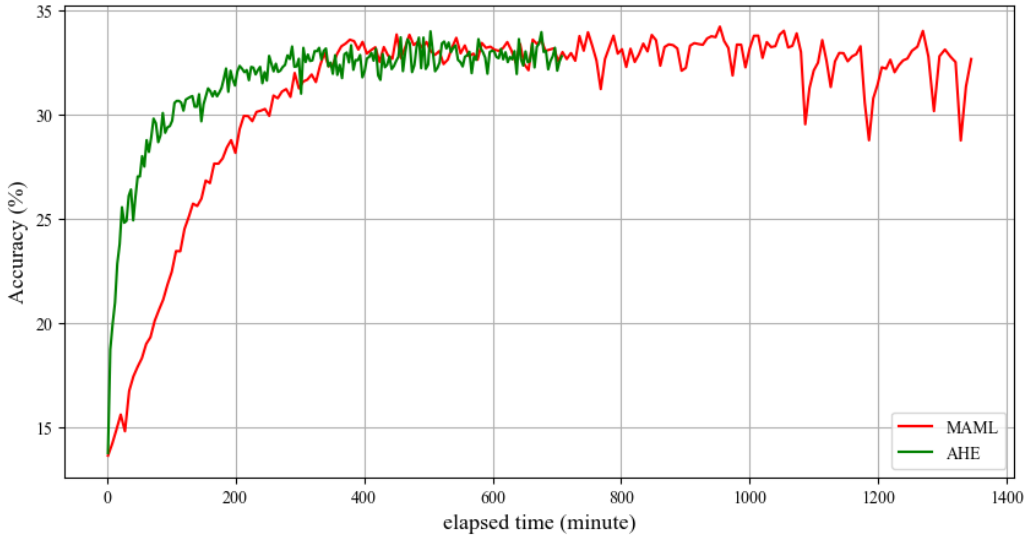



**Figure 5.** Test accuracy comparison of AHE and MAML on the MiniImagenet dataset in the 20-way 5-shot scenario across 20,000 training iterations.

### *5.3. Additional Experiment Details*

In this section, we elucidate how the approach was implemented in the aforementioned experiment, demonstrating its utility in facilitating observations and adjustments throughout the experimental process.

#### 5.3.1. Adjustment of AHE Gradient Magnitude

To create a more robust algorithm when applied to various model architectures, diverse data, and tasks, the gradient $\nabla_{\mathcal{W}_{AHE}} \mathcal{L}_{GS}$ used to update AHE in the inner loop can be normalized as

$$\mathbf{g}_{AHE} = \frac{\varepsilon \cdot \nabla_{\mathcal{W}_{AHE}} \mathcal{L}_{GS}}{\frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} |(\nabla_{\mathcal{W}_{AHE}} \mathcal{L}_{GS})_{ij}|}, \tag{16}$$

where dividing by the average absolute value and multiplying by $\varepsilon$ the average absolute value of the gradient components will be proportional to $\varepsilon$. This facilitates setting the learning rate for AHE. Moreover, as the inner loop progresses, the importance of the approximated value increases, while the actual loss value tends to decrease. Normalizing the gradient to a consistent scale addresses this issue.

#### 5.3.2. Methods for Scheduling $\varepsilon$

Various methods can be employed to schedule both $\varepsilon$ and $\tilde{\varepsilon}$. As seen in Equation (12), we schedule $\tilde{\varepsilon}$ while using a value for $\varepsilon$ that is dependent on it. Given that $\tilde{\varepsilon}$ should start from a value less than 1 and gradually approach 0, a cosine function can be utilized for scheduling. This is because learning typically progresses rapidly in the early stages and slows down as it approaches convergence. Consequently, $\tilde{\varepsilon}$ is determined as

$$\tilde{\varepsilon}_i = \frac{\cos^{-1}(2 \times (\mathrm{i} + 1) \div T - 1)}{\pi}, \tag{17}$$

where $T$ is the total number of training iterations.

## 6. Conclusions

In this work, we introduced a first-order-based Model Agnostic Meta-Learning (MAML) approach that employs a gradient similarity loss, integrating both cosine similarity loss and L2 loss. For generalizability, we maintained consistent hyperparameters across all N-way K-shot classification problems. However, there may be potential for enhanced performance by fine-tuning the hyperparameters specific to each problem. Our method proves to be competitive, especially when considering that previous first-order-based meta-learning algorithms often relied on iterations until convergence as a substitute for second-order calculations. While these methods may technically be first-order, they can be time-consuming during each loop, particularly for contemporary models with vast numbers of parameters. In contrast, our strategy involves just one additional update after each inner and outer loop, addressing the time-consuming nature of convergence-based approaches. Meta-learning offers a promising avenue to advance deep learning due to its data efficiency and adaptability across various domains employing deep learning. It also presents solutions to challenges in federated learning and online learning [32,33]. Given the benefits of meta-learning and the need to address second-order complexity, further research in this area is crucial. Beyond few-shot image classification, it would be intriguing to explore the potential performance improvements of applying AHE in reinforcement learning and regression tasks. Additionally, instead of directly utilizing the parameter values of AHE, experimenting with the outcomes derived from activation functions such as sigmoid or tanh, which exhibit a curved shape, might be anticipated to yield favorable results.

**Author Contributions:** Conceptualization, J.-H.T. and B.-W.H.; methodology, J.-H.T. and B.-W.H.; software, J.-H.T.; validation, J.-H.T. and B.-W.H.; formal analysis, J.-H.T.; investigation, J.-H.T.; resources, B.-W.H.; data curation, J.-H.T.; writing—original draft preparation, J.-H.T.; writing—review and editing, J.-H.T.; visualization, J.-H.T.; supervision, B.-W.H.; project administration, B.-W.H.; funding acquisition, B.-W.H. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was supported by the Chung-Ang University Research Scholarship Grants in 2022, the Korea government (MSIT: IITP-2021-0-01341, Artificial Intelligence Graduate School Program, Chung-Ang University), and the National Research Foundation of Korea: NRF-RS-2023-00251366.

**Data Availability Statement:** The datasets presented in this study are openely available in http://yann.lecun.com/exdb/mnist/ and https://www.kaggle.com/datasets/arjunashok33/miniimagenet.

**Conflicts of Interest:** The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

## References


1. Vinyals, O.; Blundell, C.; Lillicrap, T. Matching networks for one-shot learning. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 3630–3638.
2. Hospedales, T.M.; Antoniou, A.; Micaelli, P.; Storkey, A. Meta-Learning in Neural Networks: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 5149–5169. [CrossRef] [PubMed]
3. Huisman, M.; van Rijn, J.N.; Plaat, A. A survey of deep meta-learning. *Artif. Intell. Rev.* **2021**, *54*, 4483–4541. [CrossRef]
4. Achille, A.; Lam, M.; Tewari, R.; Ravichandran, A.; Maji, S.; Fowlkes, C.C.; Soatto, S.; Perona, P. Task2Vec: Task Embedding for Meta-Learning. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019.
5. Wu, Z.; Shi, X.; Lin, G.; Cai, J. Learning Meta-class Memory for Few-Shot Semantic Segmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021.
6. Santoro, A.; Bartunov, S.; Botvinick, M.; Wierstra, D.; Lillicrap, T. Meta-learning with memory-augmented neural networks. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016.
7. Lee, K.; Maji, S.; Ravichandran, A.; Soatto, S. Meta-Learning With Differentiable Convex Optimization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019.
8. Finn, C.; Abbeel, P.; Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017.
9. Ravi, S.; Larochelle, H. Optimization as a model for few-shot learning. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
10. Yuan, Y.; Zheng, G.; Wong, K.; Ottersten, B.; Luo, Z. Transfer Learning and Meta Learning-Based Fast Downlink Beamforming Adaptation. *IEEE Trans. Wirel. Commun.* **2020**, *20*, 1742–1755. [CrossRef]
11. Khadka, R.; Jha, D.; Hicks, S.; Thambawita, V.; Riegler, M.A.; Ali, S.; Halvorsen, P. Meta-learning with implicit gradients in a few-shot setting for medical image segmentation. *Comput. Biol. Med.* **2022**, *143*, 105227. [CrossRef]
12. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2019**, arXiv:1810.04805.
13. Gu, J.; Wang, Y.; Chen, Y.; Cho, K.; Li, V. Meta-Learning for Low-Resource Neural Machine Translation. *arXiv* **2018**, arXiv:1808.08437.
14. Li, B.; Gan, Z.; Chen, D.; Aleksandrovich, D.S. UAV Maneuvering Target Tracking in Uncertain Environments Based on Deep Reinforcement Learning and Meta-Learning. *Remote Sens.* **2020**, *12*, 3789. [CrossRef]
15. Zhang, C.; Bengio, S.; Hardt, M.; Recht, B.; Vinyals, O. Understanding deep learning requires rethinking generalization. *Commun. ACM* **2021**, *64*, 107–115. [CrossRef]
16. Li, Z.; Zhou, F.; Chen, F.; Li, H. Meta-SGD: Learning to learn quickly for few-shot learning. *arXiv* **2017**, arXiv:1707.09835.
17. Nichol, A.; Achiam, J.; Schulman, J. On first-order meta-learning algorithms. *arXiv* **2018**, arXiv:1803.02999.
18. Triantafillou, E.; Zemel, R.; Urtasun, R. Few-shot learning through an information retrieval lens. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 2255–2265.
19. Singh, R.; Bharti, V.; Purohit, V.; Kumar, A.; Singh, A.K.; Singh, S.K. MetaMed: Few-shot medical image classification using gradient-based meta-learning. *Pattern Recognit.* **2021**, *120*, 108111. [CrossRef]
20. Rajeswaran, A.; Finn, C.; Kakade, S.M.; Levine, S. Meta-learning with implicit gradients. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 113–124.
21. Zhou, P.; Yuan, X.; Xu, H.; Yan, S.; Feng, J. Efficient meta learning via minibatch proximal update. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 1534–1544.
22. Kedia, A.; Chinthakindi, S.C.; Ryu, W. Beyond Reptile: Meta-Learned Dot-Product Maximization between Gradients for Improved Single-Task Regularization. In *Findings of the Association for Computational Linguistics: EMNLP 2021*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2021.
23. Bai, Y.; Chen, M.; Zhou, P.; Zhao, T.; Lee, J.; Kakade, S.; Wang, H.; Xiong, C. How important is the train-validation split in meta-learning? In Proceedings of the International Conference on Machine Learning, Online, 18–24 July 2021; pp. 543–553.
24. Fan, C.; Ram, P.; Liu, S. Sign-maml: Efficient model-agnostic meta-learning by signsgd. *arXiv* **2021**, arXiv:2109.07497.
25. Falato, M.J.; Wolfe, B.; Natan, T.M.; Zhang, X.; Marshall, R.; Zhou, Y.; Bellan, P.; Wang, Z. Plasma image classification using cosine similarity constrained convolutional neural network. *J. Plasma Phys.* **2022**, *88*, 895880603. [CrossRef]
26. Tao, Z.; Huang, S.; Wang, G. Prototypes Sampling Mechanism for Class Incremental Learning. *IEEE Access* **2023**, *11*, 81942–81952. [CrossRef]
27. Griewank, A. Some bounds on the complexity of gradients, Jacobians, and Hessians. In *Complexity in Numerical Optimization*; World Scientific: Singapore, 1993; pp. 128–162.
28. Rusu, A.A.; Rao, D.; Sygnowski, J.; Vinyals, O.; Pascanu, R.; Osindero, S.; Hadsell, R. Meta-learning with latent embedding optimization. *arXiv* **2018**, arXiv:1807.05960.

29. Munkhdalai, T.; Yu, H. Meta networks. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; Volume 70, pp. 2554–2563.
30. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
31. Arnold, S.M.; Mahajan, P.; Datta, D.; Bunner, I.; Zarkias, K.S. learn2learn: A library for Meta-Learning research. *arXiv* **2020**, arXiv:2008.12284.
32. Fallah, A.; Mokhtari, A.; Ozdaglar, A. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 3557–3568.
33. Finn, C.; Rajeswaran, A.; Kakade, S.; Levine, S. Online meta-learning. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 1920–1930.